Package Delivery System

20181625 남현준

1. E-R model 설계에 관한 설명

A. Entity설정에 관한 설명

i. Customer Entity

프로젝트에서 구현한 database system에 우선적으로 필요한 정보는 해당 서비스를 이용하는 사용자에 관한 정보이다. 이를 위해 가장 먼저 든 생각이 사용자의 정보를 담는 Customer entity였다.

Customer entity의 attribute로는 Security number(주민등록번호), 서비스 사용자의 name(이름), phone number(전화번호), 그리고 사용자의 Account ID(계정 ID)를 선언해주었다. 이 중에서, Security number는 모든 사람에게 주어지는 고유의 번호이고, 즉 해당 값을 이용하여 모든 사용자를 식별할 수 있기에 해당 attribute를 primary key로 지정했다. 이름의 경우, 중복되는 이름을 가지는 경우가 종종 있기에 primary key로는 적합하지 않고 사용자의 부가적인 정보 부분에 해당되기 때문에 attribute로 선언하였다. Phone number attribute의 경우에는 multivariable, 즉 여러 개의 값을 가진다는 의미의 { } 표시를 해주었다. 이는, 사용자가 경우에 따라서 2개 이상의 전화번호를 사용하는 경우가 있을 수도 있기 때문에 이러한 선택을 하였다. 그 외에, 사용자가 사용하는 계정에 대한 정보는 customer entity에서 선언하는 것보다 추가적으로 account에 관한 entity를 만들어 주어 관리하는 것이 이번 프로젝트의 설명에 조금 더 맞는 것 같았다. 따라서, customer entity에서는 account ID를 foreign key로 설정하여 account에 대한 정보를 조금 더 다양하게 취급할 수 있게 하였다.

Customer entity는 account, package, shipment manager entity와 관계를 형성하게 된다. Customer_account relationship에서 customer는 1:1의 cardinality를 가지게 된다. 1:1의 cardinality인 이유는 일반적으로 하나의 사람에게 하나의 계정을 허용하기 때문이다. 중복의 계정을 허락할 수도 있겠지만 그럴 경우 보안적인 문제나 데이터를 관리하는 측면에서 매우 비효율적인 접근이라는 판단을 했기 때문에 본 프로젝트에서는 customer에게 허용되는 최대 계정의 수는 1개라는 가정 하에 모델을 진행했다.

Customer_package relationship에서 customer entity는 1:N의 cardinality를 가지게 된다. 서비스를 이용하게 될 때, 한 명의 이용자가 서비스를 여러 번 사용해서 물품을 여러 번 회사에 맡기는 일이 있을 수도 있다. 따라서 해당 relationship에서 customer는 one-

to-many의 관계를 형성하게 될 것이고, 이에 따라 relationship cardinality 또한 1:N이 될 것이다.

Shipment manager와의 relationship에서는 한 명의 customer가 여러 shipment manager와 계약을 맺을 수 있고, 마찬가지로 한 명의 shipment manager도 여러 customer와 계약을 맺을 수 있다. 따라서, 해당 관계는 many-to-many이며 N:N으로 표기될 것이다.

Customer entity는 다른 entity에서 referencing을 하는 referenced entity이기 때문에 foreign key는 존재하지 않는다.

ii. Account entity

이어서 나오게 된 entity는 account entity이다. 해당 entity에서는 Account ID(계정 ID), account name(서비스 상에서 이용하는 이름, 닉네임)등을 attribute로 선언하였다. 해당 entity에서는 account ID가 primary key로 선언되었다. Account ID는 대부분의 서비스에서 그러하듯, 중복된 계정을 허용하지 않는다. 따라서, entity에 들어올 수 있는 모든 tuple들에 대해 고유하게 식별을 가능하게 해주는 key이다. 그 외에 account entity에 선언한 attribute들은 계정정보와 관련된 다양한 정보를 저장할 수 있도록 연관지었다. 기본적으로 서비스를 사용하게 될 때, 계정 ID를 생성하게 되면 해당 서비스에서 이용할 별명 등을 정하게 된다. 그런 점을 생각해봤을 때, 서비스 이용 시 사용하는 이름의 정보를 attribute 선언하여 관리하게 되면 조금 더 데이터가 직관적으로 다가올 수 있겠다는 생각이 들어 해당 entity에 attribute를 선언했다.

Account entity는 customer entity와 customer_account의 관계를 형성하게 된다. 이 때, 한 명의 사용자는 하나의 account만 가지게 된다. 이를 ER model 상에서 표현하면 one-to-one relationship을 가진다는 의미이므로 relationship을 나타내는 다이아몬드에서 entity를 나타내는 직사각형을 향하는 화살표로 이어지게 된다. 이 때의 cardinality를 표현하게 되면 1:1이 된다.

Payment와 형성하게 되는 account_payment에서 하나의 account에서 여러 번의 payment가 조회될 수 있다. 한 사람이 서비스를 여러 번 이용하는 것으로 이런 가능성이 있는 것이다. 따라서, 해당 relationship에서 account가 one, payment가 many인 relationship이 만들어지게 되고 이를 cardinality로 표현하면 1:N이 된다.

Account entity는 customer와 연관되어 있다. 설계한 모델에서 account entity는 customer entity의 primary key를 참조해서 foreign key로 가지게 된다. 이렇게 할 경우, account의 데이터를 customer와 연결해서 쉽게 조회하고 수정하는게 가능해진다. 또한 이 경우, account의 정보에서 foreign key인 security number를 조회했을 때, 해당 값이

customer entity에 존재하지 않는 정보라면 데이터베이스에 오류가 생겼다는 것을 알 수 있게 되고 이를 통해 참조 무경성을 유지할 수 있게 된다.

iii. Payment entity

Payment entity는 어떤 형식의 transaction—돈의 지불—이 일어났을 때의 정보를 저장하기 위해 선언한 entity이다.

해당 entity의 primary key로는 payment ID를 선언하였고 그 외에 attribute로 payment date, payment amount, payment method, payment status등을 선언하여 transaction에 관련된 정보를 추가적으로 저장할 수 있도록 했다.

Payment는 account, package entity와 relationship을 형성하게 된다.

Account와 형성하게 되는 account_payment에서의 relationship cardinality는 account가 one, payment가 many인 one-to-many, 즉 1:N으로 표시된다. 한 사용자가 여러 번의 서비스를 위해 돈을 지불할 수 있지만 하나의 지불정보는 한 사용자로부터만 나올 수 있다. 따라서, 해당 관계에서는 특정 사용자의 정보를 알 수 있어야 참조 무결성이 성립되고 이를 위해 account entity의 primary key인 account ID를 foreign key로 가지게 된다.

Package와 형성하게 되는 package_payment에서의 relationship cardinality는 payment가 one, package가 one인 one-to-one, 즉 1:1으로 표기된다. 하나의 지불정보에 대해 한 package가 관련되어 있을 것이며 마찬가지로 한 package도 하나의 지불 정보만을 가지게 될 것이기 때문이다. 따라서 해당 관계에 대한 정보를 알기 위해서는 어떤 지불이 해당 package에 이용됐는지 알 수 있어야 참조 무결성이 성립한다. 해당 관계에서는 package가 payment에 대한 정보를 알 수 있어야 함으로 package가 payment를 referencing 할 것이고 이에 따라 package entity에 foreign key가 생기게 될 것이다.

iv. Package entity

사용자의 정보와 더불어 빠뜨릴 수 없는 정보가 서비스를 이용하며 택배로 보내게 되는 물품에 관한 정보이다. 물품에 관한 정보를 저장하기 위해 우선 물품에 관한 정보를 관리할 수 있는 package entity를 선언해주었다.

Package entity에서 각 물품들은 개별적으로 식별이 가능한 상태여야 한다. 이를 위해 서비스를 통해 배달하고자 하는 물품에 대해 고유의 번호를 매기는 것이 바람직하다고 판단되었고 이를 위해 package ID attribute를 선언하였다. 해당 attribute는 package entity에 소속되는 모든 tuple들을 개별적으로 식별하기 위해 사용될 것이므로 entity의 primary key로 설정될 것이다. 가이드라인에서 제시한 조건들을 보았을 때, 해당 물품이

우편인지, 작은 박스인지 혹은 큰 박스인지에 관한 정보를 저장할 필요가 있었다. 따라서, package entity의 attribute로 type of package를 선언해주어서 해당 entity에 정보가 저장될 때 어떤 종류의 택배인지 알 수 있게 하는 것이 맞다고 생각했다. 그 다음에는 일반적으로 택배를 생각했을 때 떠오르는 특성들인 무게, 가격 등을 일단 해당 entity의 attribute로 넣어주었다.

Package entity는 customer, shipment, payment, recipient와 관계를 형성하게 된다. Customer와 형성하게 되는 customer_package relationship에서는 한 명의 customer가 다수의 package를 회사에 부탁할 수 있다. 따라서 customer와 package 사이의 관계는 package가 many, customer가 one인 many-to-one의 관계가 되고, 이를 나타내면 N:1이 된다.

Shipment와 형성하게 되는 relationship에서, shipment는 출발지와 목적지가 같은 package들을 한 곳에 모아서 출발하는 package의 묶음이라고 생각할 수 있다. 따라서, 한 shipment에 여러 package가 대응되고, 이를 cardinality로 나타내면 package가 many, shipment가 one, 즉 1:N(혹은 N:1)로 나타낼 수 있다.

Payment와 형성하게 되는 package_payment에서의 relationship cardinality는 payment entity에서도 설명했듯이 one-to-one, 즉 1:1으로 표기된다. 해당 관계에서는 package가 payment에 대한 정보를 알 수 있어야 함으로 package가 payment를 referencing 할 것이고 이에 따라 package entity는 payment의 primary key인 payment ID를 foreign key로 가지게 될 것이다.

Recipient와의 관계에서, package는 해당 recipient의 정보가 유효한지 알 수 있어야 한다. 따라서, recipient의 primary key인 recipient ID를 foreign key로 가지게 되며 이를 통해 참조 무결성을 유지할 수 있을 것이다.

또한, Package entity는 account entity를 참조한다. 그렇기 때문에 package entity는 account의 primary key인 account ID를 foreign key로 가지게 된다. Account ID를 foreign key로 설정함을 통해 package와 account간의 상관관계를 쉽게 알 수 있게 된다. 만약, package의 account ID가 account entity의 tuple들과 match 되지 않는다면 데이터베이스에 문제가 생겼다는 것을 의미하고 참조 무결성 또한 유지되지 않는다.

v. Recipient entity

Recipient는 package가 배송됐을 때 해당 package를 전달받는 사람에 대한 정보를 저장하게 된다. Primary key로 Recipient ID를 선언했으며 attribute로 address를 선언하여 해당 recipient의 주소를 알 수 있게 하였다.

Recipient는 package와 relationship을 형성하게 된다. 해당 관계에서 한 명의 recipient는 여러 개의 package를 받을 수 있고 하나의 package는 한 명의 recipient만 존재할 것이기 때문에 해당 관계는 recipient가 one, package가 many인 one-to-many, 즉 1:N의 cardinality를 가지게 될 것이다.

Recipient는 package가 referencing하는 referenced entity이기 때문에 foreign key는 존재하지 않을 것이다.

vi. Information entity

프로젝트 명세서에 few points to consider 부분을 보면 대부분의 경우, 회사는 어떤 물품이 배달되는지 관심을 가지지 않지만 몇몇 특수한 케이스들이 있다고 적혀 있다. 해당 조건을 생각해보면, 배달되는 물품에 관한 정보를 저장할 수 있는 entity 역시 필수불가결하다고 생각되었다. 해당 정보를 관리할 수 있는 방법은 두 가지가 있었다. 하나는 package 내부에 추가적으로 attribute를 선언해 택배에 들어있는 물품이 어떤 물품인지, 또 어떤 용도로 사용할 것인지에 대해 저장하는 것이고 다른 하나는 package를 identifying entity로 가지는 weak entity를 선언해 정보를 관리하는 것이다. 처음에는 package에 대한 정보를 한 entity 안에서 관리하려고 했지만 points to consider에 나오는 hazardous materials인지 아닌지, international shipment인지 아닌지를 판별할 때, package entity에서만 관리하게 되면 attribute의 값으로 null이 제법 많이 포함될 것으로 보였다. 이러한 생각을 한 이유는, 해당 정보는 필요할 수도 있고 아닐 수도 있기 때문이다. 프로젝트 명세서를 보면 회사는 기본적으로 어떤 물품이 운송되는지에 대해 신경 쓰지 않고 몇몇 특별한 경우에만 해당 정보를 추적 및 관리를 한다고 되어있기 때문이다. 따라서 꼭 필요한 경우가 아니라면 null이 들어가도 무방하다는 뜻이기도 하다. 경우에 따라 null을 허용하는 것이 좋은 경우도 있지만, 일반적으로 null값을 배제할 수 있는 방법이 있다면, 해당 방법을 선택하는 것이 좋은 디자인 방법이다. 따라서, package를 identifying entity로 가지는 information을 weak entity로 선언하여 정보의 모순과 정확성이 떨어지는 것을 방지하는 것이 좋겠다는 판단을 내리게 되었다.

Weak entity의 primary key는 identifying set의 primary key이다. 따라서, foreign key이자 identifying key는 package의 primary key인 package ID가 될 것이다. 또한, weak entity인 경우, 관계되어 있는 entity의 cardinality를 따라가게 된다. 이 경우, information entity는 1:N의 cardinality를 가지게 될 것이다.

vii. Shipment entity

그 다음에는 package들을 모아서 운송하게 되는 shipment entity를 정의하였다. 해당

entity의 primary key로는 shipment ID를 선언하였다. 모든 shipment에 개별적으로 ID를 선언함으로 모든 shipment를 식별하는 것이 가능해지기 때문이다.

요구하는 query에 따르면 현재 운송되고 있는 화물의 위치와 배송지연 등의 여부를 추적할 수 있어야 한다. 해당 요구사항을 충족하기 위해서는 언제 물품이 회사에 의해서 배송이 시작되었는지, 현재 화물의 위치, 예상 배달 시간 등을 데이터베이스를 통해 확인할 수 있는 방법이 있어야 한다. 해당 사항들은 shipment를 묘사하는 특성들이지만, shipment entity 내부에 attribute들을 추가해서 관리하기에는 정보의 비일관성, 비효율적으로 작동한다는 생각이 들었기 때문에 단일 entity 보다는 추가적으로 관련 entity를 생성하여 relationship을 통해 관계를 표현해주는 것이 낫다고 판단했다. 이러한 과정을 통해 생성한 entity는 delivery와 location이 있다.

Shipment entity는 여러 entity들과 relationship을 형성한다. 후에 서술할 delivery entity와 형성하는 shipment_delivery는 shipment의 배송에 대한 relationship이다. 한 개의 shipment는 one delivery에 대응하기 때문에 one-to-one relationship이고 이를 나타내면 1:1가 된다.

후에 서술할 transportation entity와 형성하게 되는 shipment_transportation relationship에서 한 개의 shipment는 여러 개의 transportation과 relationship을 형성할 수 있다. 한 개의 shipment에 대해서, transportation의 종류는 여러 가지가 있을 수 있다. 따라서, shipment와 transportation 사이의 relationship은 shipment가 one, transportation이 many인 one-to-many (or many to one)가 되고 이를 나타내면 1:N (or N:1)으로 표현할 수 있다.

Shipment는 shipment manager와 형성하는 relationship에서 shipment가 many, shipment manager가 one인 many-to-one의 cardinality를 가지게 된다. 따라서, Shipment는 어떤 manager가 담당하는지에 대한 정보를 알 수 있어야 한다. 따라서, shipment manager와 형성하는 relationship에서 shipment가 shipment manger를 referencing 하게 되고 이로 인해 shipment는 shipment manager의 primary key인 manager ID를 foreign key로 가지게 될 것이다.

viii. Delivery entity

Delivery는 특정 택배의 운송에 관해 정의된 attribute를 가지고 있는 entity이다. Delivery entity에는 tracking number, picked up date, estimated delivery date, timeliness of delivery, delivery status, actual arrival time 등이 있다. 해당 entity의 primary key로는 tracking number, 즉 운송장 번호를 생각했다. 운송장 번호는 우리가 일상 생활에서 택배의 배송을 조회할 때 사용되는 번호이고 모든 택배에 일괄적으로 부여되는 고유의 번호

이기 때문에 해당 entity의 tuple들을 식별하는 것에 문제가 없을 것이다. 요구사항의 points to consider 부분을 보면 언제 배송을 시작했는지와 그에 맞는 예상 도착 시간을 알 수 있어야 한다고 적혀있다. 따라서, delivery entity에 picked up time/date와 estimated delivery time/date를 선언해주어 해당 정보를 데이터베이스에서 관리할 수 있도록 attribute를 선언했다. Query 부분에서는 예상 배송시간보다 늦게 배송이 완료된 택배들을 찾을 수 있기를 원했는데, 해당 query를 위해 delivery entity에 delivery status와 actual arrival time을 선언하였다. Delivery status attribute를 통해 해당 택배가 배달되고 있는 중인지, 아니면 배송이 완료된 상태인지를 확인할 수 있을 것이고 actual arrival time attribute를 통해 배송이 완료됐을 때의 시간/날짜를 저장한 뒤 해당 값을 estimated delivery time/date와 비교하여 배송이 지연되었는지 아니면 제때 도착하였는지 판단할 수 있게 될 것이다.

Delivery는 shipment, location과 relationship을 가지게 될 것이다.

Shipment와 형성하는 shipment_delivery에서 하나의 shipment는 하나의 delivery 정보를 가지게 될 것이고, 하나의 delivery 또한 하나의 shipment에 대응되는 정보를 가질 것이다. 따라서 해당 관계는 one-to-one, 1:1이 될 것이다.

Location과 형성하는 delivery_location에서 하나의 delivery는 화물이 이동하면서 위치가 갱신될 수 있고, 하나의 location은 언제나 하나의 delivery의 위치를 알려줄 것이다. 따라서, 해당 relationship은 delivery가 one, location이 many인 one-to-many, 1:N이 될 것이다.

Shipment_delivery에서 delivery는 배송에 대한 정보가 유효한 shipment에 대해서 저장되고 있는지 알 수 있어야 한다. 따라서, delivery는 shipment를 referencing하게 될 것이고 이에 따라 delivery는 shipment의 primary key인 shipment ID를 foreign key로 가지게 될 것이다.

ix. Location entity

배송상황에서 같이 알고 싶은 정보는 해당 택배가 어디 쯤에 있는지에 관한 정보일 것이다. 해당 정보를 관리하기 위해 location entity를 추가할 필요를 느꼈다. 해당 entity를 선언할 때, 지명이 모든 장소를 유일하게 식별할 수 있는 것처럼 보이기 때문에 처음에는 지명과 주소를 primary key로 설정하려고 생각했다. 하지만 조금 더 고민을 해보니 같은 지명을 가지는 장소가 있을 수도 있고, 이와 같은 경우에 지명만을 select하는 query가 요청될 경우 정보의 불확실성이 있을 수도 있겠다는 생각이 들었다. 우리나라의 경우에도 지번을 생각해보면 같은 지번이지만 상세주소로 들어가보면 아파트의 동, 호에 따라서 최종 주소가 달라지게 된다. 따라서, 비슷한 방식으로 본 프로젝트에서 구현한

location entity의 경우, location ID에 일정 지역을 유일하게 식별할 수 있는 지번과 같은 것을 할당하고, 상세 주소에 해당되는 부분을 location name과 address 등의 entity를 추가적으로 선언하는 것으로 관리하겠다는 결정을 했다.

Location entity는 delivery, transportation entity와 relationship을 형성하게 된다. Location_delivery relationship에서 location은 delivery와 many-to-one의 관계가 된다. 현재 진행중인 배송들에 대해 여러 배송이 한 location에서 대기중이라면 같은 좌표에 위치하고 있을 수도 있기 때문이다. One-to-many의 관계를 나타내면 1:N (or N:1)으로 표현할 수 있다. 이 때, delivery의 경우, 모든 상황에서 위치의 정보를 도출할 수 있어야 요구서에 적혀 있는 package의 위치 정보를 알아낼 수 있는 방법에 대한 solution을 제공할 수 있을 것이다.

Location은 delivery와 relationship을 형성하게 되는데, 이 때 유효한 delivery에 대해서 location의 정보를 저장하고 있는지 알 수 있어야 한다. 따라서, location과 delivery 사이의 relationship에서, location이 delivery를 referencing 하게 되고 이에 따라 location은 delivery의 primary key인 tracking number를 foreign key로 가지게 될 것이다.

x. Transportation entity

Shipment가 배송되기 위해서는 배송 수단이 필요할 것이다. 그에 관한 정보를 저장하기 위해서 transportation이라는 entity를 선언하였다. 해당 entity에서는 배송에 사용되는 수단에 대한 정보를 관리하기 위해 transportation ID, type of transportation, capacity, transportation status 등을 attribute로 선언했다. Transportation ID는 해당 수단을 유일하게 식별해줄 수 있는 번호를 의미한다. Type of transportation과 같이 얘기를 해보자면, 운송수단으로 사용될 수 있는 것에는 트럭, 비행기, 선박 등이 있을 수 있고 여기서 말하는 transportation ID는 트럭의 번호판, 비행기의 항공기 등록번호, 선박의 선박 번호 등을 의미하게 될 것이다. 사용될 수 있는 수단이 여러가지이기 때문에 ID와 transportation type을 묶어서 primary key로 설정해야 하나 라는 생각이 있기도 했지만 결국에 transportation ID 자체가 개별적으로 부여되는 독립적인 번호이기 때문에 transportation ID만으로 모든 개체를 독립적으로 식별할 수 있겠다는 판단을 내렸다.

Transportation은 shipment, shipment manager entity와 relationship을 형성하게 된다.

Transportation_shipment relationship에서 하나의 shipment는 여러 개의 transportation 중 하나가 선택되어서 배송이 진행되고, transportation은 하나의 shipment를 배송한다고 가정한다. 이 가정을 따른다면 shipment가 one, transportation이 many인 one-to-many의 cardinality를 가지게 될 것이다. 해당 관계에서, transportation은

유효한 shipment가 적재되었는지 확인할 수 있어야 한다. 따라서, transportation은 shipment를 referencing하게 되며 이에 따라 shipment의 primary key인 shipment ID를 foreign key로 가지게 될 것이다.

Shipment manager와의 relationship에서 transportation은 관리하는 다수의 관리자가 있을 것이고, manager 또한 여러 개의 transportation을 관리할 수 있을 것이다. 해당 관계를 cardinality로 표현하게 되면 many-to-many, N:N이 된다.

xi. Shipment manager entity

Shipment manager entity는 Shipment를 배송할 때, 해당 shipment를 책임지는 관리자에 관한 정보를 저장하는 entity이다. 해당 entity는 manager ID, name, affiliation, phone number 등을 attribute로 선언했다. Manager ID는 shipment를 담당하는 사람의 식별번호, 고유의 사원번호 등을 저장하는 attribute, 즉 해당 entity의 primary key가 될 것이다. 이외의 다른 attribute들은 담당자의 추가적인 정보들을 저장하게 될 것이다.

해당 entity는 customer, shipment와 relationship을 형성하게 된다.

Customer와의 relationship에서는 한 명의 customer가 여러 shipment manager와 계약을 맺을 수 있고, 마찬가지로 한 명의 shipment manager도 여러 customer와 계약을 맺을 수 있다. 따라서, 해당 관계는 many-to-many이며 N:N으로 표기될 것이다.

Shipment와 형성하는 shipment_manager에서 한 명의 manager는 다수의 shipment를 관리할 수 있지만 하나의 shipment는 한 명의 manager만 존재할 것이다. 따라서 해당 relationship의 cardinality는 shipment manager가 one, shipment가 many인 one-to-many, 1:N가 될 것이다.

해당 entity는 referenced인 entity이므로 foreign key는 따로 존재하지 않을 것이다.

2. Schema diagram에 관한 설명

A. Customer entity

Customer entity는 (1)의 ER model 설명에서 얘기했듯이 Security number, name, phone number를 entity의 attribute로 가지게 된다. ERwin에서 해당 entity를 구현할 때, primary key를 security number로 설정해주고 나머지는 일반 attribute로 선언해서 entity를 구성했다. 해당 attribute들의 logical data type은 integer, varchar, char로 구성했다.

Integer의 경우 primary key인 security number의 logical data type으로 선언했다. Char와 integer 사이에서 고민했지만 security number의 경우 특별한 경우가 아니라면 모든자리가 숫자로 이루어져 있기 때문에 integer로 선언해도 문제가 없을 것이라고 판단됐다. 만약 security number에 숫자 외에 다른 문자가 들어간다는 가정이 있었다면 char형을 logical date type으로 선언했을 것 같다.

Name의 경우, varchar을 logical data type으로 선언했다. Char로 해도 무방하지만 char로 할 경우, 만약 char(18)을 logical data type으로 사용했고 실제로 들어온 이름 길이가 18자가 되지 않는다면 그 만큼의 메모리가 낭비되는 것이기 때문에 최대글자 제한이 있되 그 만큼의 문자가 들어오지 않으면 자동으로 크기가 바뀌는 varchar이 조금 더 나은 logical date type이라고 생각했다.

Phone number 같은 경우에 우리나라에서 집전화는 더 이상 쓰이지 않고 대부분 가입을 하게 될 때 휴대전화 번호를 해당 정보를 등록할 것이기 때문에 name고 다르게 char형으로 선언을 하더라도 무방할 것이라는 생각이 들었다.

해당 entity의 attribute들은 phone number를 제외하면 모두 null 값을 허용하지 않기로 설정했다. 이런 판단을 한 이유는 핸드폰 번호의 경우에는 간혹 가다가 없는 경우가 있을 수도 있지만, primary key인 security number와 name의 경우, null 값을 허용하지 않는 속성이거나 모든 사람들이 가지고 있는 속성일 것이기 때문이다.

Customer entity의 경우 Account, package, shipment manager와 관계를 형성한다.

Customer_Account relationship의 경우 ER model에서 one-to-one relationship을 가진다고 설명을 했고 해당 부분을 schema diagram으로 나타낼 경우, account와 customer를 non-identifying 기능을 이용해서 연결한 뒤, 해당 간선의 속성의 cardinality properties 부분에서 cardinality value를 1로 설정해주었다. 이렇게 하면 non-identifying을 통해 one-to-one 관계라는 것을 ERwin에서 표현할 수 있다.

Customer_Package relationship의 경우 customer가 one, package가 many의 cardinality를 가지는 관계가 형성된다. 이를 표현하기 위해서 마찬가지로 Erwin의 non-identifying 기능을 이용하여 customer entity를 먼저 클릭하고, package entity를 클릭해서 해당 관계를 표현해주었다.

Customer_manager relationship은 many-to-many의 relationship을 형성하게 되고 이를 표현하기 위해 Erwin의 many-to-many 기능을 이용하여 해당 entity들을 연결시켰다.

해당 entity는 다른 entity를 referencing 하지 않기 때문에 foreign key가 생기지

않았다.

B. Account entity

Account entity는 account ID, account name을 attribute로 가지게 된다. Entity를 구성할 때 account ID의 primary key 부분을 체크해주었고 나머지 부분들을 기입했다. 해당 entity의 logical data type으로는 varchar을 사용했다.

Account ID, account name은 varchar형의 logical data type을 사용했다. Account ID와 account name의 경우, 대부분의 제공되는 서비스들이 그러하듯 최대 글자수 제한이 있고 최대 글자수보다 작게 쓰는 경우가 많기 때문에 일반 char형을 쓰게 된다면 메모리의 낭비로 이어질 수 있다. 따라서 해당 attribute는 varchar이 쓰는 것이 설계 상 조금 더 좋아보인다는 생각이 들었다.

해당 entity에서 primary key인 account ID와 account name 모두 null 값을 허용하지 않도록 하였다. 일반적으로, 계정을 생성하게 될 때, account name과 account id는 필수적으로 기입해야 하는 정보이기 때문이다.

Customer entity와 생성하게 되는 관계에서 account entity는 customer entity의 primary key인 security number를 foreign key로 갖게 된다. 이 말은, customer entity에 특정 tuple이 새롭게 입력될 때, 해당 tuple의 security number가 customer entity 내부에 존재하지 않으면 잘못된 접근을 의미하게 된다.

Account entity는 customer, payment entity와 relationship을 형성하게 된다. Account entity에서도 설명했듯이 해당 모델은 한 사용자가 하나의 계정만을 생성할 수 있다는 전제 하에 진행되었다. 따라서, account와 customer가 형성하는 relationship은 one-to-one이라는 전제이고 Erwin의 non-identifying을 사용해서 account와 customer entity를 연결하고 cardinality value를 1로 설정해주었다.

Payment와의 관계에서 한 명의 사용자가 account를 이용해서 여러 번의 서비스에 대해 돈을 지불할 수 있다. 따라서, 해당 관계는 account가 one, payment가 many인 one-to-many로 나타낼 수 있을 것이다. 이것을 구현하기 위해 Non-identifying을 이용해서 account와 payment를 연결하였다.

Account entity에서 account는 특정 이용자가 생성했을 때 사용할 수 있는 것이기 때문에 customer entity에서 해당 이용자가 있는지 없는지 확인할 수 있어야 한다. 따라서 해당 entity는 customer entity를 referencing하여 customer entity의 primary key인 security number를 foreign key로 가지게 된다.

C. Payment entity

Payment entity는 payment ID, payment date, payment amount, payment method, payment status 등을 attribute로 가지게 된다. 이 중, payment의 정보를 유일하게 식별하기 위해 payment ID를 primary key로 설정하였다.

Attribute의 logical data type으로는 integer, datetime day to minute, varchar 등이 사용되었다.

Payment ID와 payment amount는 숫자와 관련된 정보이므로 integer를 통해 정보를 관리하는 것이 가능하다.

Payment date는 해당 payment가 이루어진 날짜를 저장하는 attribute이다. 따라서, 해당 payment가 일어난 날짜와 조금 더 구체적으로 알 수 있게 분 단위로 나타낼 수 있으면 좋을 것 같아서 datetime date to minute형으로 data type을 선언하였다.

그 외의 정보들은, 문자열과 관련되어 있고 최대 글자수를 선언하는 varchar형을 통해 해당 글자수보다 적게 들어온다면 자동적으로 남은 공간만큼의 memory를 할당해제하는 varchar형이 메모리의 관리성 측면에서 좋다는 판단이 들어 해당 data type을 사용하게 되었다.

해당 entity는 null 값을 허용하지 않도록 설정했다. Payment가 발생하게 되었을 때, 일반적으로 언제, 어떻게, 처리의 여부에 대한 정보는 필수적으로 저장되어야만 후에 잘못된 정보가 저장되었을 때 해당 정보의 삭제 여부를 판단할 수 있기 때문이다.

Payment는 package, account와 relationship이 생기게 된다.

Payment_package에서 하나의 payment는 하나의 package에 대해 발생하게 되는 비용에 대한 relationship을 정의한다. 따라서, 해당 관계는 one-to-one으로 표현될 것이다. 이 것을 위해 payment와 package를 Erwin의 non-identifying으로 연결해준 뒤, cardinality value를 1으로 설정해주었다.

Payment_account에서 payment와 account는 one-to-many의 relationship을 가지게 될 것이다. 한 명의 사용자가 account를 이용해서 여러 번의 서비스에 대해 돈을 지불할 수 있기 때문이다. 해당 relationship은 Erwin의 non-identifying을 이용해 entity들을 연결해주었다.

Payment는 어떤 account에 의해서 지불이 진행되었는지에 대한 정보를 알 수 있어야 한다. 따라서, payment_account에서 payment는 account를 referencing하게 되며 이에 따라 acccount의 primary key인 account ID를 foreign key로 가지게 될

것이다.

D. Package entity

Package entity는 package ID, type of package, weight, price를 attribute로 가지게 된다. 이 중, package ID를 primary key로 설정하고 나머지는 일반 attribute가 된다. 해당 entity에서는 integer와 varchar logical data type을 사용했다.

Attribute 중에서 type of package만 varchar logical data type을 가지고 나머지는 전부 integer data type을 가지게 된다. 이렇게 설정한 이유는 설명서에 type of package는 flat envelope, small box, larger boxes중에 하나가 될 것이라고 적혀있는데 이 경우, char을 이용하게 되면 글자수에 따라 메모리의 낭비가 생길 수 있기 때문에 varchar형을 사용하는 것이 조금 더 바람직한 설계라고 생각했다. 그 외의 attribute들은 전부 integer의 형태로 나타내어지는 정보들이기 때문에 integer형으로 정보를 저장하는 것에 문제가 없을 것이라 판단됐다.

Package entity가 customer entity와 non-identifying 관계로 연결되면서 customer의 security number가 foreign key로 넘겨지게 된다. Package의 경우 배송을 부탁한 사람이 있을 것이고 연관지어서 해당 인물에 관한 정보를 찾을 수 있어야 하기 때문에 foreign key를 통해 customer entity를 referencing 하도록 설계를 하였다. 이 때의 cardinality는 many-to-one이고, Erwin에서 zero, one, more으로 설정했다. 한 사용자가 1개 이상의 package를 맡겼을 수도 있고 하나도 맡기지 않은 상태일 수도 있기 때문이다.

Package와 shipment가 형성하는 relationship은 non-identifying 기능을 이용했다. 이 때, 한 shipment는 출발과 도착이 같은 여러 개의 package가 존재할 수 있기 때문에 many-to-one의 relationship이 형성된다. 따라서, cardinality를 나타내게 되면 zero, one, or more이 되게 된다. 이 관계에서 package가 어떤 shipment에 있는지 알 수 있어야 하기 때문에 foreign key로 shipment ID를 shipment entity에서 reference 할 필요가 있다.

Package와 payment가 형성하는 relationship은 non-identifying을 이용해 연결해준 뒤 cardinality value를 1로 설정해주었다. 해당 relationship에서 package는 어떤 payment에 의해 비용이 지불되었는지 알 수 있어야 하고 이에 따라 payment의 primary key인 payment ID를 foreign key로 가지게 된다.

Package와 recipient사이의 relationship은 recipient가 많은 package를 배송받을 수 있고, package는 한 번에 하나의 recipient만 가지기 때문에 package가 one, recipient가 many인 one-to-many relationship이다. 따라서, package는 recipient의

정보가 유효한지 판단할 수 있어야 하고 이에 따라 recipient의 primary key인 recipient ID를 foreign key로 가지게 될 것이다.

Package entity에서는 foreign key인 security number를 포함한 모든 attribute에서 null 값을 허용하지 않도록 설정했다. Package ID와 security number는 null 값을 허용하게 된다면 특정 사용자가 package를 맡겼다는 사실에 모순이 생길 수도 있기 때문에 허용하지 않도록 설정했다. Shipment ID의 경우, 아직 shipment에 적재되지 않은 상태일 수도 있기 때문에 null 값을 허용하도록 하였다. 나머지 attribute들의 경우에는 현실에서도 서비스를 이용하기 전에 무게, package의 종류 등을 종합해서 가격을 매기는 식으로 진행되기 때문에 null 값을 허용하지 않는 방향으로 설정했다.

E. Recipient entity

Recipient는 recipient ID를 primary key로 가지고 추가적인 attribute로 address를 가지게 된다. Recipient ID의 경우, 물건을 받게 되는 사람의 신원 확인 수단, 즉 수령자의 정보에 해당되는 security number 같은 것이 될 것이다.

해당 entity의 logical data type은 integer와 varchar로 선언했다. Primary key인 recipient ID를 security number라고 가정하고 entity를 구성했기 때문에 logical data type으로 integer를 사용하였고, address의 경우, 최대 글자 수를 선언하되 최대 글자보다 적은 입력이 들어온다면 할당된 메모리의 크기를 줄여줄 수 있는 varchar이 조금 더 구현에 적합해보였기 때문이다.

해당 entity는 null을 허용하지 않도록 결정했다. Recipient의 정보를 primary key인 recipient ID는 말할 것도 없고, address의 경우 location, delivery entity와 연계해서 recipient의 주소에 예상 시간 내에 도착했는지 확인할 것이기 때문이 null이 들어오게 된다면 해당 검색이 불가능해질 것이기 때문이다.

해당 entity는 package와 relationship이 생기는데 이 때, package에서 recipient entity를 referencing하기 때문에 foreign key가 따로 생기지는 않는다. 해당 relationship은 recipient가 one, package가 many인 one-to-many의 relationship이다. ER model에서도 설명했듯이, 한 명의 recipient가 다량의 package를 받는 것이 가능하기 때문이다.

Recipient는 referenced되는 relationship만 있기 때문에 foreign key가 생기지 않는다.

F. Information entity

해당 entity는 package의 추가적인 정보를 저장하게 되는 entity이므로 package를 identifying set으로 가지게 된다. 이를 위해 information과 package를 Erwin의 identifying 기능을 이용해서 연결해주었다. 해당 relationship에서의 cardinality는 zero, or one이다. 있을 수도 있고 없을 수도 있는데 있다면 하나의 package에 대해 하나의 추가정보가 있을 수 있기 때문이다.

Information entity는 weak entity이기 때문에 연결된 entity의 primary key를 identifying key로 가지게 된다. Discriminator로는 type of product, product name, purpose of shipment 등이 있다.

Logical data type으로는 varchar을 이용했다. 해당 entity에 들어가는 데이터들이 종류, 이름, 이유 등에 관련된 것들인데 이러한 정보들은 사람들에 따라 다르게 묘사할 수도 있기 때문에 char을 통해 고정된 메모리를 할당하는 것보다 varchar을 통해 최대 몇 자까지 적을 수 있다로 허용하는 것이 메모리 적인 측면에서 봤을 때 효율적으로 보였다.

Identifying key인 package ID를 제외하면 모두 null 값을 허용했는데 이렇게 판단한 이유는 기본적으로 해당 information을 의무적으로 회사에서 관리해야 할 필요가 없고 특수한 경우에만 해당 information에 대해 필수적으로 관리 및 추적해야 하기 때문이다. 따라서 일반적인 경우에는 해당 정보가 들어오지 않아도 상관 없기 때문에 null 값을 허용하도록 설계했다.

G. Shipment entity

Shipment entity는 shipment ID를 primary key로 가지게 된다. 해당 attribute의 logical data type으로는 integer를 적용했다.

Shipment entity는 package, delivery, transportation, manager entity들과 관계를 형성하게 된다.

앞서 말한 package entity와의 관계는 one-to-many의 관계가 될 것이고 이를 Erwin을 통해 표현하면 zero, one, or more이 될 것이다.

Shipment와 delivery는 one-to-one relationship을 형성하게 된다. 하나의 shipment는 delivery에 관한 정보를 하나만 가질 수 있기 때문이다. Erwin의 non-identifying 기능을 이용해서 delivery entity와 연결해주었다. 이 때, 해당 간선의 cardinality value를 1로 설정해주어서 one-one의 relationship임을 알 수 있게 했다.

Shipment와 transportation은 one-to-many의 relationship을 형성한다. 하나의 shipment는 여러 수단을 통해 이동될 수 있지만 하나의 수단은 한 번에 하나의 shipment만 배송할 수 있기 때문이다. Non-identifying을 이용해서 연결해주는 것을 통해 one-to-many임을 나타내었다.

Shipment와 shipment manager의 관계에서 shipment가 many, shipment manager가 one인 관계가 형성된다고 ER model에서 설명하였다. 해당 관계는 one-to-many이고 Erwin의 non-identifying을 이용해서 나타내었다.

Shipment는 shipment를 담당하는 사람의 정보를 알 수 있어야 한다. 따라서, shipment manager와 형성하는 관계에서 shipmen manager를 referencing하게 되고 이에 따라 shipment manager의 primary key인 manager ID를 foreign key로 가지게 된다.

H. Delivery entity

Delivery entity는 tracking number를 primary key로 가지고 picked up time/date, estimated delivery time/date, timeliness of delivery, delivery status, actual arrival time/date를 attribute로 가지게 된다.

해당 entity의 logical data type으로는 integer, datetime day to minute, varchar을 사용했다. Primary key인 tracking number는 integer 형으로 선언했다. 운송번호는 대부분의 경우 숫자로만 나타내기에 integer형을 사용하는 것이 가장 효율적으로 보였기 때문이다. Picked up time/date, estimated delivery time/date, actual arrival time/date의 경우 시간과 관련되어 있는 attribute들이고 택배의 배송을 조회할 때, 분 단위 까지 표기하는 것이 조회하는 것에 과하지 않고 적당한 선이라고 느꼈다. 따라서 해당 값들은 datetime day to minute형을 사용해서 정보를 저장하도록 하였다. 그 외 timeliness of delivery나 delivery status 같은 경우는 배송 상황이나 어느 시간대에 배송되는 지에 대한 정보이므로 문자열에 관한 attribute가 된다. 따라서, 앞선 entity들에서도 그랬던 것 같이 varchar형을 logical data type으로 주어 최대 글자 수만큼 사용하지 않았다면 자동적으로 사용된 메모리를 글자수에 맞게 해주는 동작을 통해 메모리를 효율적으로 쓸 수 있게 했다.

해당 entity에서는 null값을 허용하는 쪽으로 판단을 내렸다. Delivery를 tracking하며 delivery status와 actual arrival time/date에 대한 정보 또한 저장되게 되는데, 만약 delivery status가 아직 배송중이라면, actual arrival time/date에 들어갈 수 있는 정보가 없기 때문이다.

Delivery entity는 shipment, location entity와 relationship을 형성한다.

Shipment_delivery entity는 shipment와 delivery가 형성하는 relationship에 대한 정보를 알려준다. 해당 relationship에서 특정 shipment의 배송 여부에 대한 정보를 얻을 수 있게 되는데, 이 때 한 shipment에 대해서는 하나의 delivery 밖에 존재할 수 없다. 따라서, 해당 relationship은 one-to-one (1:1)이 되며 Erwin에서 non-identifying을 이용해서 연결해준 뒤, cardinality value를 1로 설정해주었다. Delivery는 특정 shipment에 대한 배송 상황에 관한 데이터를 저장하기 때문에 어떤 shipment인지를 알 수 있는 방법이 필요하다. 따라서 해당 관계를 통해 shipment entity의 primary key인 shipment ID를 foreign key로 가지게 된다.

Delivery_location은 배달이 진행될 때, 어느 위치에 있는지에 대한 정보를 알려주는 relationship이다. Delivery에 대해서 location은 이동을 하며 변하기 때문에 여러 location이 한 delivery에서 사용될 수 있다. 따라서 해당 relationship은 one-to-many이며 Erwin에서 non-identifying을 이용해 연결해준 뒤 cardinality를 one, or more로 설정해주었다. One, or more인 이유는 모든 delivery는 시작과 동시에 최소 하나의 location을 가지기 때문이다.

I. Location entity

Location entity는 Location ID를 primary key로 가지고 location name, address를 attribute로 가지게 된다. Location ID의 logical data type으로는 integer를 사용했고 location name과 address는 varchar data type을 사용했다.

해당 entity는 null을 허용하지 않도록 했다. 위치를 조회할 때, 세부정보도 조회할 수 있어야 하기 때문이다 .

Location은 delivery와 relationship을 형성한다.

Delivery와 형성하게 되는 relationship 앞서 G에서 설명한 것과 같이 one-to-many의 relationship이다. 이 때의 cardinality는 non-identifying set을 이용한 one, or more로 set된다. 해당 relationship에서, location은 어떤 delivery에 대한 데이터를 저장할 것인지 알 수 있어야 한다. 따라서, location은 delivery를 referencing 하게 되고 이로 인해 delivery의 primary key인 tracking number를 foreign key로 가지게 된다.

J. Transportation

Transportation은 transportation ID를 primary key로 가지고 type of transportation, capacity, transportation status, shipment ID를 추가적인 attribute로

가지게 된다. 해당 attribute들은 transportation ID와 capacity만 integer logical data type으로 설정하고 나머지는 varchar형으로 설정해주었다.

해당 entity는 null 값을 허용하지 않도록 결정했다. 배송수단의 경우, null 값이 허용될 경우, 정확한 수단을 조회하기 어렵기 때문에 null을 허용하지 않도록 했다.

Transportation은 shipment, shipment manager와 relationship이 생기게 된다.

Shipment와 만들게 되는 relationship에서 하나의 shipment는 여러 개의 transportation 중 하나에 적재되어 배송이 시작된다. 따라서, 해당 relationship은 shipment가 one, transportation이 many인 one-to-many의 relationship이다. 이를 나타내기 위해서 non-identifying 기능을 이용해 shipment와 transportation을 연결했고 cardinality를 zero, one or more로 설정해주었다. Zero, one or more인 이유는 해당 관계에서 shipment가 적재되지 않는 transportation도 있을 수 있기 때문이다.

Shipment manager와 형성하게 되는 relationship은 many-to-many의 관계이다. 해당 설정의 이유로는 한 manager에게 할당될 수 있는 transportation의 수는 하나 이상이고, 마찬가지로 한 transportation을 담당하게 되는 manager의 수 또한 한 명이상일 수가 있기 때문이다. 따라서 해당 relationship은 Erwin의 many-to-many 기능을 이용해서 연결해주었다.

K. Shipment manager

Shipment manager는 primary key로 manager ID를 가지고, attribute로 name, affiliation, phone number를 가지게 된다.

해당 entity의 logical data type으로는 integer, varchar을 선언했다. Integer의 경우, manager ID와 phone number에 사용될 것이다. Name과 affiliation은 varchar logical data type을 이용했다.

해당 entity에서는 affiliation을 제외하면 null 값을 허용하지 않았다. Shipment manager는 물건을 담당하는 사람이기에 연락이 가능해야 하고 primary key인 manager ID와 attribute name은 사람을 판별하는데 중요한 요소들이기 때문이다. Affiliation의 경우, 소속이 없는 프리랜서일 가능성도 있을 수 있어서 해당 부분은 null을 허용하도록 하였다.

Shipment manager는 shipment, transportation, customer와 relationship을 형성하게 된다.

Shipment와의 relationship은 manager가 one, shipment가 many인 one-to-many

의 relationship이다. 해당 표현을 위해 Erwin의 non-identifying을 사용해서 shipment manager와 shipment를 연결해주었다.

Transportation과의 relationship은 many to many이다. 다수의 manager가 다수의 transportation을 관리하는 것이 가능하기 때문이다. 해당 표현을 위해 Erwin의 many-to-many를 이용해서 relationship을 나타냈다.

Customer와의 관계는 many to many이다. 한 명의 customer은 다수의 shipment manager와 계약을 할 수 있고 마찬가지로 shipment manager도 다수의 customer와 계약을 하는 것이 가능하기 때문이다. 이를 위해 Erwin의 many-to-many 기능을 이용해 customer와 shipment manager를 연결했다.

해당 entity는 referenced만 되기 때문에 foreign key를 가지지 않는다.